Un Modèle Homogène pour la Confidentialité et l'Intégrité des Données Relationnelles

Romuald Thion, Stéphane Coulondre
romuald.thion@liris.cnrs.fr, *stephane.coulondre@liris.cnrs.fr*

LIRIS - Laboratoire d'InfoRmatique en Images et Systèmes d'information
UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1
Université Lumière Lyon 2/Ecole Centrale de Lyon,

Bâtiment Blaise Pascal (501), 20, avenue Albert Einstein,
69621 Villeurbanne Cedex, France

Résumé

Les systèmes d'information sont devenus omniprésents dans les organisations, et ne sont plus cantonnés à un nombre limités d'applicatifs : qu'il s'agissent de données d'exploitation, de rapports confidentiels ou d'une devanture accessible sur Internet, les systèmes d'information s'étendent dans toutes les branches des organisations. La sécurité est aujourd'hui un des enjeux considérables des systèmes, que se soit vis-à-vis de la productivité, des montants financiers, de l'image de marque ou des aspects légaux. Avec la mise en ligne des systèmes et l'augmentation du nombre d'utilisateurs, et donc d'attaquants potentiels y compris internes, il faut définir, mettre en place et administrer des politiques de contrôle d'accès définissant qui a droit à quoi dans les systèmes.

Le contrôle d'accès à base de rôle est apparu ces dernières années comme un standard *de facto*. Souple et intuitif, il simplifie l'administration des grands systèmes. Malheureusement, les spécificités de ce modèle les plus intéressantes ne sont pas encore intégrées de façon formelle et homogène dans les bases de données relationnelles, pierre angulaire des systèmes d'information. Cet article propose d'utiliser les notions de dépendances de données pour représenter et intégrer dans les systèmes de gestion de base de données relationnelles les politiques de contrôle d'accès à base de rôles avec leurs fonctionnalités les plus intéressantes, telles que les notions de hiérarchies ou de contraintes. Mettant à profits les travaux sur les dépendances, notre approche permet d'assister la conception des politiques et de s'assurer de leurs consistances. Une validation expérimentale montre l'intérêt et l'expressivité de la proposition. La fertilisation croisée entre contrôle d'accès et dépendances ouvre des perspectives nombreuses et intéressantes : administration de politiques distribuées, prise en compte du temps et de la mobilité des utilisateurs ou encore définition de nouvelles règles de contrôle d'accès.

Mots-clefs

Sécurité et Confidentialité, Logique, Base de Données Relationnelle, Dépendances Génératrices de Tuples Contraintes, Contrôle d'Accès RBAC.

1 Introduction

Les Systèmes de Gestion de Bases de Données (SGBD) sont les pierres angulaires des Systèmes d'Information (SI) : ils sont utilisés par tout type d'application et sont la technologie incontournable de stockage, gestion et interrogation des données. La corruption ou l'utilisation frauduleuse de ces systèmes ne concerne alors pas qu'une seule application ou seulement quelques utilisateurs, mais peut affecter l'ensemble des SIs. De plus, la mise en ligne des SIs et l'augmentation du nombre d'applications Web ont considérablement augmenté le nombre de menaces et d'attaquants potentiels, même si une grande partie des attaques (si ce n'est la majorité) provient directement de l'intérieur des organisations. Que ce soit des données d'exploitation courantes ou d'ordre stratégiques, la sécurité du SI passe par la sécurité des SGBDs.

La prise de conscience des risques liés à la sécurité informatique incite les organisations à mettre en place des politiques de sécurité : un ensemble de règles et de mesures visant à garantir la sécurité du SI. Le Contrôle d'Accès (CA) est un des mécanismes incontournables dans la définition et l'établissement d'une politique de sécurité. Une politique de CA définit « qui a droit à quoi » dans le SI, des mécanismes mettant en application la dite politique. Le CA vise principalement à garantir la confidentialité de l'information (l'information ne doit être accessible qu'aux ayant droits : protection contre la lecture non autorisée), son intégrité (l'information doit être cohérente et valide : protection contre l'écriture non autorisée) et indirectement sa disponibilité (l'information doit être accessible par les ayant droits).

Le CA est un domaine de recherche très actif où de nombreuses proposition de modèles ont été faites [FKC03, BS05]. Le modèle à base de rôle (Role-Based Access Control - RBAC) [SCFY96] a été largement étudié, étendu et appliqué : les travaux comportent entre autres un standard [FSG+01], des spécifications XML [OAS05], des modèles d'administration [SM99, Cra03], l'étude des contraintes temporelles [JBLG05] ou encore des méthodes d'analyse [LT04, KPP03]. Le modèle RBAC est aujourd'hui largement répandu comme en témoignent ses nombreuses implémentations dans la santé ou le commerce électronique par exemple [FKC03].

Actuellement, les SGBDs (mais aussi les architectures de services Web, les outils d'administration de sécurité) supportent les politiques RBAC, mais malheureusement, ces implémentations des travaux sur RBAC sont assez limitées et n'intègrent qu'une version réduite de ce modèle de CA [BS05]. Nous pensons que les SGBDs ne profitent pas assez des nombreuses avancées effectuées dans ce domaine de la sécurité : elles se limitent à des politiques RBAC peu flexibles, qui ont du mal à intégrer certaines spécificités des organisations car déconnectées des problématiques métier, sans outil d'aide à la conception et avec peu de possibilités de vérification.

Nous pensons qu'il faut intégrer les avancées du domaine du CA dans les BDs tout en respectant les spécificités du modèle relationnel. Il est admis que les modèles de CA intégrés dans les BDs doivent être exprimés dans le modèle logique de données [BS05] : c'est à dire en terme de relations, d'attributs et de tuples. A notre connaissance, les travaux ne proposent pas assez de mettre à profit les recherches effectuées dans le domaine des BDs pour répondre le plus correctement possible aux problèmes actuels posés par la gestion du CA.

Notre proposition d'intégration et de vérification de politiques RBAC dans les BDs est basée sur l'utilisation des travaux sur les *dépendances de données*. Cette approche nous permet de capitaliser l'existant dans le domaine des BDs : les dépendances que nous utilisons [MS96] sont parmi les plus générales qui soient. Ce cadre théorique des dépendances est formellement bien défini et s'intègre parfaitement au sein du modèle relationnel : nous ne rajoutons pas de concepts extérieurs aux BDs. Les outils que nous manipulons peuvent être à la fois utilisés pour les aspects de sécurité que nous présentons ici, ou pour des usages plus traditionnels tels que la conception de schémas relationnels, les calculs de couvertures ou l'optimisation des requêtes. Cette fertilisation croisée entre les dé-

pendances, originellement développées pour assurer l'intégrité des données, et le CA va nous permettre de proposer un cadre homogène pour la conception de politiques de CA et s'assurer de leur intégrité (ex : pour vérifier que la politique est bien cohérente ou pour vérifier si des propriétés sont effectivement respectées par la politique). L'expressivité du cadre formel utilisé nous permettra de développer un nouvel ensemble de contraintes spécifiques au CA et de proposer des extensions prenant en compte des caractéristiques actuellement difficiles à intégrer.

Suite à cette introduction, nous présenterons les travaux relatifs au CA dans les BDs et nous détaillerons les apports de notre approche. La section 3 présentera le cadre des dépendances que nous utilisons. Notre proposition sera développée dans la section 4 où nous indiquerons comment structurer les politiques RBAC dans les BDs et comment concevoir et vérifier de telles politiques. La section 5 illustrera notre proposition par une validation expérimentale. Enfin la dernière section conclura et proposera des perspectives de recherches.

2 Sécurité Dans les Bases de Données

Les recherches en BD se sont depuis longtemps attachées à garantir la sécurité des données. La notion d'intégrité a particulièrement abouti, donnant naissance aux contraintes d'intégrité, il s'agissait alors de s'assurer que les données stockées étaient cohérentes et valides. L'avènement des BDs comme pierre angulaire des SI, gérant des données touchant à tous les niveaux d'une organisation, et l'augmentation du nombre et de la variété des utilisateurs des systèmes a nécessité de contrôler les accès aux données : que les utilisateurs n'aient accès qu'aux informations nécessaires à la bonne réalisation des tâches qui leurs incombent. Il a fallu définir les autorisations et imposer « qui a droit à quoi » : la problématique du CA.

2.1 Contrôle d'Accès Dans les Bases de Données

Contrôler les accès, c'est déterminer si un sujet (un processus, une machine, un utilisateur, ...) peut effectuer une opération (sélectionner, supprimer, modifier, créer, ...) demandée sur une ressource (une tuple, une table, un objet, un fichier, ...). Une permission est un droit d'opération sur une ressource. Une politique de CA spécifie les droits des sujets sur les ressources d'un système, dans le but de renforcer la politique de sécurité de l'organisation. Les modèles historiques de CA que sont les modèles discrétionnaires et mandataires ont été les premiers mis en oeuvre dans les SGBDs. Bertino et Sandhu ont récemment écrit un état de l'art exhaustif de la sécurité dans les BDs [BS05], dont une partie importante traite du CA dans les BDs relationnelles.

Il est indiqué, comme dans [FKC03], que les modèles discrétionnaires (DAC - Discretionnary Access Control - les usagers peuvent définir les permissions d'accès aux données dont ils sont les propriétaires) ont une faiblesse importante : ont ne peut plus contrôler ce qui est fait de l'information une fois que celle-ci a été accédée par un utilisateur légitime. Par exemple, un utilisateur ayant accès en lecture à une ressource, peut en créer une nouvelle avec le même contenu. Il sera ainsi propriétaire de la nouvelle ressource « copie » et peut alors la rendre publique pour tout le monde, ce qui briserait complètement la confidentialité de cette donnée. Cette faiblesse rend les systèmes discrétionnaires vulnérables à de nombreuses attaques, notamment les chevaux de Troie. De plus, comme les utilisateurs disposent d'une grande liberté, ces modèles s'avèrent difficiles à administrer : les reponsables de la sécurité n'ont pas le contrôle complet des ressources, les usagers peuvent ainsi définir des permissions allant à l'encontre de la politique de sécurité.

C'est notamment pour répondre à ce type de problèmes que les modèles mandataires (MAC - Mandatory Access Control - les usagers ne peuvent pas définir les permissions d'accès, ils ne sont pas propriétaires des données) et multi niveaux sont apparus. Ils sont basés sur une

classification des données (par exemple : Non-classé, Confidentiel, Secret, Très Secret). Ce type de modèle est bien adapté aux applications militaires qui sont très rigides, ou pour tout environnement clos et contrôlé (par exemple, les systèmes bancaires). Les SGBDs ayant utilisé les modèles mandataires n'ont eu que peu de succès commercial, vraisemblablement à cause de leur rigidité et de la hiérarchisation stricte qu'ils imposent aux utilisateurs. Une formalisation en logique utilisant les notions de dépendances a été proposée par Cuppens et Gabillon pour les BDs multi-niveaux [CG99]. Cet article aborde des problèmes issus de la classification des données tels que la poly-instanciation ou la gestion des leurres (cover story), dont [Cup00] propose une synthèse exhaustive. Nos travaux tâche de suivre cette approche pour les BDs relationnelles mais pour les modèles de CA à base de rôles.

2.2 Contrôle d'Accès à Base de Rôles

RBAC est basé sur la constatation qu'une grande partie des décisions de CA sont déterminées par l'autorité hiérarchique ou la fonction du sujet dans son organisation [SCFY96, FKC03] : cela forme le concept central de rôle. L'introduction de ce concept dans les politiques de CA comme « intermédiaire » entre les sujets et les permissions facilite et simplifie grandement les tâches d'administration. Dans ce modèle, on n'affecte jamais directement les permissions aux utilisateurs, tous les privilèges sont attribués par l'intermédiaire des rôles. Le modèle RBAC a été introduit pour combler les déficits des modèles mandataires et discrétionnaires qui s'avèrent trop rigides, trop complexes à administrer ou invérifiables.

Les rôles peuvent être perçus comme des collections de permissions. Lorsqu'un utilisateur a besoin d'effectuer une tâche, il suffit de lui attribuer le ou les rôles correspondants. Ainsi, quand un utilisateur change de fonction dans son organisation, il suffit de changer les rôles qui lui sont attribués sans introduire directement de notion de révocation de permissions. La figure 1 illustre l'organisation de RBAC. Les acronymes URA et PRA signifient respectivement User-Role Assignment et Permission-Role Assignment.

En sus de ces principes de bases, le modèle RBAC a introduit la notion de *hiérarchie de rôles* et celle de *contraintes*[1] dans le CA. La notion de hiérarchie permet de réduire le nombre de regroupement de permissions et d'affectation de rôles aux utilisateurs en introduisant une notion d'héritage entre les rôles : un rôle hérite des permissions accordées à ses parents. La notion de contrainte a été introduite pour représenter les spécificités des organisations et limiter les privilèges des usagers du système. Nous détaillerons ces notions en section 4.

Bertino et Sandhu attachent de l'importance au modèle RBAC dans les SGBDs [BS05]. Il apparaît comme une innovation importante dans le domaine du CA, sa flexibilité et la simplification de l'administration qu'il apporte a su séduire éditeurs et organisations. Dans ce modèle, la définition des droits d'accès est basée sur la notion de rôle, qui permet d'organiser la politique de sécurité au plus proche possible de la structure de l'organisation qui la déploie. Il s'agit du modèle auquel nous allons nous attacher car c'est celui qui répond au mieux aux besoins actuels des organisations : expressif, flexible, simple et calqué sur la structure l'organisation (a contrario des modèles discrétionnaires par exemple).

Malheureusement, les mises en oeuvre qui ont été faites du modèle RBAC dans les SGBDs sont limitées et ne concernent qu'une version assez simplifiée de ce qui est proposé [RS98]. Par exemple, la gestion de RBAC dans Oracle [SB98] ne traite que des affectations utilisateur-rôle et permission-rôle mais pas de la hiérarchie des rôles. Il a été proposé d'utiliser les mécanismes de triggers pour combler

[1] Nous attirons ici l'attention sur la polysémie du mot *contrainte*. Il peut désigner des *relations entre variables et constantes* (par exemple, $X \leq Y, X \geq 2 \times Z + 1, 3 = T, 2 \neq 3$), il peut aussi s'agir de *restrictions sur les concepts du modèle RBAC* (par exemple, aucun utilisateur ne peut endosser les rôles r_1 et r_2), de *contraintes d'intégrité* au sens de *dépendances de données*, ou au sens de *contraintes que doit vérifier toute politique RBAC*. Afin d'éviter les confusions, nous préférons utiliser dans cet article les termes de *contraintes arithmétiques*, *contraintes métier*, *dépendances* et *contraintes d'intégrité de politique*

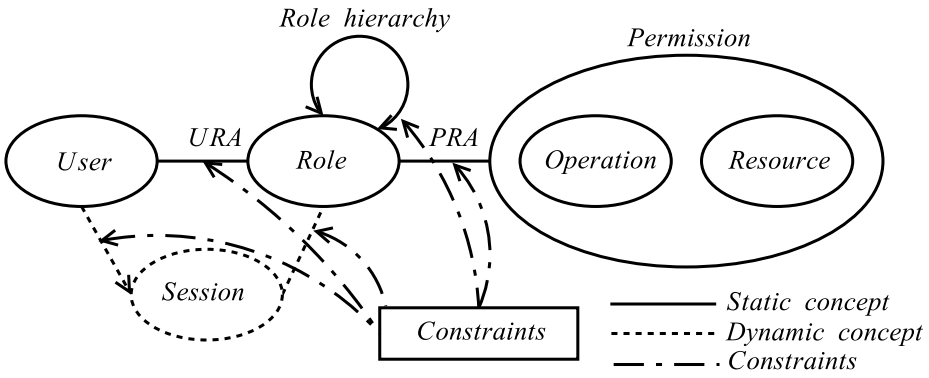

FIG. 1 – Le modèle RBAC [SCFY96, FKC03], pas de formalisme spécifique

cette lacune [BD03], mais cela ne nous semble pas être une bonne idée car cette proposition repousse les problématiques de la gestion des politiques de CA à des problèmes techniques de vérification de cohérence et de mise à jour de code. Ceci tend à alourdir et à rendre pénible l'administration des politiques. Cette approche ne semble pas être la meilleure possible car elle se base sur des mécanismes ajoutés au modèle relationnel (triggers), et non pas intégrés à ce dernier.

2.3 Panorama des Travaux sur les Politiques à Base de Rôles

De nombreux travaux ont été proposés sur les politiques RBAC. Des auteurs s'attachent à des implémentations concrètes du CA dans les SGBDs, d'autres se consacrent à l'étude du CA de façon plus théorique, plus détachée de l'implémentation qui en sera faite. Par exemple, des auteurs définissent un cadre formel pour l'expression et la validation de toute une gamme de politiques de CA [BCFP03]. Pour ce qui est des modèles à base de rôles plus spécifiquement, on trouve la définition d'un standard [FSG+01] et des spécifications XML [OAS05]. Le standard NIST existant spécifie les fonctionnalités a implémenter pour gérer des politiques RBAC, le profil XACML pour RBAC définit quant à lui un format d'échange.

Un aspect des plus détaillés concernant le modèle RBAC est celui des contraintes. Les contraintes sont un ensemble de restrictions devant être garanties dans la politique. Il s'agit notamment de garantir la séparation des tâches et d'éviter qu'un utilisateur dispose de trop de permissions. Les contraintes sont un aspect important du modèle RBAC, voire même une des principales motivations [AS00]. Les auteurs de [GB98] ont également abordé la notion de contraintes entre utilisateurs et rôles et entre rôles, aboutissant à un ensemble de propriétés devant être respectées par la politique RBAC pour que les contraintes soient cohérentes. L'article [Bar02] étudie l'intégration du modèle RBAC, y compris la notion de hiérarchie de rôles, dans les BDs déductives, mais les notions de contraintes n'ont pas été intégrées. Il est dommageable que les implémentations de RBAC dans les SGBDs ne prennent pas en compte un des arguments forts du modèle RBAC.

Un autre aspect concerne les extensions du modèle RBAC, notamment en ce qui concerne les aspects temporels [JBLG05]. Le modèle GT-RBAC permet la spécification d'une large variété de contraintes temporelles dans la politique de CA. Pareillement, ces avancées n'ont pas trouvé d'implémentation dans les SGBDs faute de paradigmes de BD adéquats à leur représentation et à leur manipulation.

Globalement, même si la nécessité d'un cadre formel rigoureux pour la représentation et la vérification de politiques de CA est admise, les résultats proposés ne semblent pas toujours tirer profit de recherches effectués dans d'autres

domaines. Lorsqu'il s'agit d'intégrer le CA au sein des SGBDs, il nous paraît intéressant de profiter des travaux effectués en la matière en utilisant comme cadre pour l'expression et la validation des politiques, un cadre issu directement des recherches en BDs.

2.4 Notre Approche

Nous avons d'un côté une riche littérature sur le modèle RBAC, mais dans la pratique peu d'implémentation, ou en tout cas des réalisations limitées, certains concepts du modèle RBAC s'avérant difficiles à implémenter, comme la hiérarchie de rôle ou les contraintes. De plus, les recherches sur la gestion des politiques sont assez éloignées des problématiques des SIs, ce qui conduit quelquefois à des propositions *ad hoc*, mises en oeuvre comme surcouche des SGBDs, mais qui s'intègrent quelquefois assez mal au modèle relationnel. D'un autre côté, les recherches dans le domaine des BDs relationnelles sont fructueuses et proposent des résultats sur des aspects théoriques ou pratiques, dont la portée dépasse largement le cadre strict des dépendances. Les travaux sur les classes de dépendances de données de plus en plus expressives par exemple apportent des résultats nombreux [MS96, WTM01, Wan02, MW00, Cou03] que nous allons mettre à profit pour la sécurité. Notre approche est basée sur une fertilisation croisée entre les recherches sur les dépendances et celles sur le CA. Notre proposition s'attache à intégrer RBAC de la manière la plus homogène et la plus complète possible dans les BDs relationnelles

Nous montrons que l'expressivité apportées par les classes de dépendances telles que les *dépendances génératrices de tuples contraintes* permet de représenter tout la palette de concepts propres à RBAC qui actuellement n'est qu'en partie prise en compte. Il est intéressant de noter qu'originellement les dépendances on été introduites pour répondre à des problèmes d'intégrité de données, problèmes que notre approche lie à la confidentialité, nous sommes ainsi capables d'identifier une gamme de « contraintes d'intégrité sur les politiques » de CA.

La conception de politique est d'une grande importance pratique [BCFP03], nous pensons qu'il faut des outils facilitant la conception et la maintenance des politiques de CA. Ces outils doivent : avoir une interface graphique appropriée, être capables d'exprimer les mécanismes et les spécificités des modèles, être capables de vérifier la consistance et d'interroger les politiques [GB98] mais aussi d'avoir un mécanisme d'inférence compréhensible par les non-logiciens [HW03]. Les dépendances et leurs procédures de preuves associées nous permettent de répondre à ces critères. Nous utilisons ainsi un seul et même cadre théorique liant les notions jusqu'alors traitées indépendamment de CA et d'intégrité. La conception de la politique de sécurité et la conception des BDs sont ainsi assistées par un seul et même outil.

3 Dépendances Génératrices de Tuples Contraintes

Les contraintes d'intégrité sont des propriétés supposées être satisfaites par toutes les instances d'un schéma de BD. Les propriétés devant être satisfaites étant typiquement l'égalité ou l'existence de certaines données en fonction d'autres. La littérature a étudié des classes de contraintes d'intégrité appelés *dépendances*. Dans un but d'unification des différentes dépendances, la logique du premier ordre a été adoptée comme cadre commun car elle permet de représenter toutes les dépendances utilisées en pratique par des formules à la structure assez simple [AHV95]. La théorie des dépendances a trouvé par exemple des applications avec la conception de schémas ou l'optimisation de requêtes.

La classe de dépendances que nous utilisons, est celle des *dépendances génératrices de tuples contraintes* [MS96] (Constrainted Tuple-Generating Dependencies - CTGDs). Nous les avons choisies car elles forment un sur-ensemble de la plupart des dépendances existantes. Nous disposons ainsi d'un cadre très expressif et générique, tout en sachant que sous réserve de certaines restrictions nous pouvons nous limiter à des classes dépendances inclues dans les CTGDs

et bénéficier des résultats afférents. Dans cet article, nous utilisons le formalisme de la logique des prédicats du premier ordre pour l'expression des dépendances.

3.1 Présentation

Les *dépendances génératrices de tuples contraintes* étendent les *dépendances génératrices de tuples* [BV84, Cou03] en ajoutant des *contraintes*. Ce dernier terme *contraintes* est a entendre au sens des *bases de contraintes* [Rev95] et de la *programmation logique par contraintes* [JM94] et non pas au sens de contraintes d'intégrité (terme auquel nous préférerons celui de dépendances).

Les dépendances génératrices de tuples contraintes peuvent être formellement représentées par des expressions de la forme (1) du tableau 1 [MS96]. Il s'agit donc d'un fragment de la logique du premier ordre, ne comportant pas de disjonctions, pas de symboles de fonction, pas de négation et une seule implication par formule. Dans ce tableau, $X = X_1 \cup \ldots \cup X_n$, $Y = Y_1 \cup \ldots \cup Y_m$, $X'_n \subseteq X$, $X \cap Y = \emptyset$, p_i, q_j sont des symboles de prédicats, C est une conjonction de contraintes sur des variables de X, C' est une conjonction de contraintes sur des variables de X et de Y. Les termes de X sont quantifiés universellement. Enfin Y ne désigne non pas tous les termes de la conclusion de la CTGD, mais uniquement ceux qui ne sont pas quantifiés universellement dans l'hypothèse.

Les contraintes C et C' sont des formules exprimées dans un langage $\mathcal{L}$ sur un domaine $\mathcal{D}$. Les CTGDs sont *paramétrées* par la classe de contraintes utilisée, comme les bases de contraintes [Rev95]. Les CTGDs sont définies quels que soient $\mathcal{L}$ et $\mathcal{D}$, pourvu que la classe de contraintes soit décidable. Les domaines de contraintes les plus courants sont les égalités/inégalités sur les entiers, les rationnels, les réels ou les contraintes linéaires arithmétiques.

Des formes restreintes remarquables des CTGDs sont (notons que nous réutiliserons les acronymes CGD, NGD, TGD, TTGD et que les expressions se réfèrent au tableau 1) :

- les dépendances génératrices d'égalité (Equality-Generating Dependencies - EGD), qui généralisent les dépendances fonctionnelles, expression (2), où C est limitée à des conjonctions d'égalités,
- les dépendances génératrices de contraintes (Constraint-Generating Dependencies - CGD) [BCW95], qui généralisent les dépendances génératrices d'égalités, expression (3),
- les dépendances génératrices de nullité (Nullity-Generating Dependencies - NGD)[2], expression (4),
- les dépendances génératrices de tuples totales (Full-TGD ou Total-TGD) qui ne comportent pas de variables quantifiées existentiellement, qui généralisent les dépendances multi-valuées, expression (5),
- les dépendances génératrices de tuples ou dépendances généralisées (Tuple-Generating Dependencies - TGD [BV84, Cou03]), qui ajoutent des variables quantifiées existentiellement aux TTGDs, expression (6).

Maher et Srivastava illustrent l'expressivité des CTGDs par l'exemple 1 du tableau 1 [MS96], qui exprime le fait que « tout employé (nom, chef, sécurité) avec une accréditation basse a *au moins* un chef qui dispose d'une accréditation moyenne ». Cette dépendance ne peut pas être réduite à une combinaison de dépendances génératrices de tuples ou génératrices de contraintes, en effet la CGD de exemple 2 du tableau 1, imposerait que *tous* les chefs d'un employé aient une accréditation moyenne.

Les CTGDs ont été développées pour représenter de nouveaux types de relations entre les données qui apparaissent avec les nouveaux paradigmes de BD, tels que les bases de contraintes, les BDs déductives, spatiales ou temporelles. Par exemple, pour des données géographiques, on utiliserait un domaine de contraintes exprimant des relations topologiques ou spatiales entre les objets, comme X *contient* Y ou X *est au nord est de* Y. Les CTGDs ont été étendues par les dépen-

[2]Pour la représentation en pratique des NGD par les CTGDs, nous utiliserons à la place de $\perp$ une contrainte qui ne peut jamais être satisfaite, comme $1 \neq 1$

(1)	CTGD	$\forall X\ p_1(X_1), \ldots, p_n(X_n), C(X) \rightarrow \exists Y\ q_1(X'_1, Y_1), \ldots, q_m(X'_m, Y_m), C'(X, Y)$
(2)	EGD	$\forall X\ p_1(X_1), \ldots, p_n(X_n) \rightarrow C(\tilde{X})$
(3)	CGD	$\forall X\ p_1(X_1), \ldots, p_n(X_n), C(X) \rightarrow C'(\tilde{X})$
(4)	NGD	$\forall X\ p_1(X_1), \ldots, p_n(X_n), C(X) \rightarrow \bot$
(5)	TTGD	$\forall X\ p_1(X_1), \ldots, p_n(X_n) \rightarrow q_1(X'_1), \ldots, q_m(X'_m)$
(6)	TGD	$\forall X\ p_1(X_1), \ldots, p_n(X_n) \rightarrow \exists Y\ q_1(X'_1, Y_1), \ldots, q_m(X'_m, Y_m)$

exemple 1	$\forall N_1, B_1, S_1\ emp(N_1, B_1, S_1), S_1 < 5 \rightarrow \exists B_2, S_2\ emp(B_1, B_2, S_2), S_2 \geq 7$
exemple 2	$\forall N_1, B_1, S_1\ emp(N_1, B_1, S_1), emp(B_1, B_2, S_2), S_1 < 5 \rightarrow S_2 \geq 7$

TAB. 1 – Expression en logique des dépendances

dances génératrices de tuples contraintes disjonctives [Wan02], qui présentent de nombreuses perspectives d'utilisation des CTGDs.

3.2 Procédures de Preuve

L'aspect théorique relatifs aux dépendances le plus étudié est certainement celui de l'*implication logique* : étant donnée un ensemble de dépendances F et une dépendance g, g est-elle satisfaite pour toute instance de schéma qui satisfait F ? Ce problème de l'implication s'avère être un problème incontournable dans tout les cas d'utilisation des dépendances, pour l'optimisation ou l'équivalence de requêtes par exemple.

Deux procédures de preuves pour les CTGDs ont été développées dans [MS96]. C'est principalement une propriété du domaine de contrainte utilisé [LM89] (Independance of Negative Constraints - INC) qui fait leur différence. D'un point de vue très général, ces procédures de preuves reprennent le *chase* de Beeri et Vardi [BV84] dédiée aux TGDs et y adjoignant la propagation des contraintes. Le *chase* est une procédure basée sur l'application successive de chacune des dépendances de F jusqu'à prouver que g est logiquement impliquée par F, noté $F \vDash g$.

Le principe du *chase* est de supposer l'existence de tuples telles que l'hypothèse l de $g \equiv l \rightarrow r$ soit satisfaite, puis de traiter F comme un opérateur de fermeture générant des tuples à partir de cette hypothèse $F(l)$. A chaque étape du calcul de cette fermeture, un magasin de contraintes stocke les différentes contraintes produites, et la condition suivante est testée :

- si $F(l)$ contient r, alors $F \vDash g$ (*cas de terminaison* 1),
- si le magasin de contraintes contient une inconsistance, alors $F \vDash g$ par vacuité (*cas de terminaison* 2),
- si $F(l)$ ne contient pas de copie de r et que l'on a atteint un point fixe, alors $F \nvDash g$ (*cas de terminaison* 3).

Cette procédure de preuve est semi-décidable : elle répondra en un temps fini pour une réponse positive, mais peut ne pas terminer pour une réponse négative, par exemple dans le cas où l'on arrive pas à obtenir de point fixe. Pour les applications de l'implication que nous proposons dans cet article, on peut techniquement limiter le nombre d'applications de dépendances pour le calcul de $F(l)$. Cette limite est à évaluer selon le nombre de dépendances que contient F pour minimiser le nombre de vrais négatifs (cas où l'algorithme est arrêté pour avoir a atteint la limite alors qu'il aurait terminé). Pour notre application, cette limite dépend notamment de la profondeur de la hiérarchie de rôles. Notons que pour certaines restrictions des CTGDs, telles que les Total-TGDs évoquées en section 3.1, le *chase* est décidable [BV84].

4 Proposition

Afin de garantir un CA fiable et de faciliter la conception des politiques, nous pensons qu'il faut se rapprocher le plus possible des paradigmes existants dans le domaine des BDs

et d'utiliser les outils développés en la matière. Cette approche nous permet de concilier la gestion des données « métier » (*i.e.* les données d'exploitation du SGBD) ainsi que la gestion du CA. Cette section décrit les principes de base du modèle RBAC, explique comment structurer les politiques dans les BDs et décrit comment les concevoir et les vérifier.

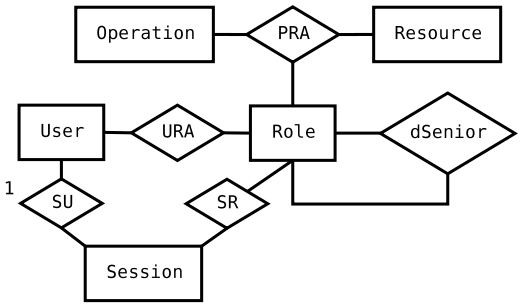

FIG. 2 – Schéma ER de la base des faits RBAC

4.1 Structuration du Contrôle d'Accès

Afin de faciliter l'intégration, la vérification ainsi que la mise en oeuvre des politiques de CA RBAC dans les BDs, nous avons choisi de structurer les différents composants entrant en jeu dans ces politiques :

- les principes d'autorisation, de hiérarchie des rôles : le coeur du modèle : F_{RBAC}. Les dépendances de cet ensemble sont toutes des Total-TGD,
- les contraintes d'intégrité qui permettent d'assurer qu'une politique est valide : $F_{integrite}$. Un ensemble de dépendances de type TGD et NGD,
- les données de la politique de CA, nommée dans la littérature « base de données RBAC » (RBAC Database) [GB98] : F_{faits}. Il s'agit des faits de la politique, un ensemble d'atomes ne comportant pas de variables,
- les contraintes « métier », *i.e.* les contraintes au sens RBAC du terme comme l'exclusion mutuelle ou les pré-requis : F_{metier}. Un ensemble de dépendances de type CTGDs.

Les ensembles F_{RBAC} et $F_{integrite}$ définissent comment fonctionne le modèle RBAC et les contraintes qui permettent de s'assurer de l'intégrité de *toute* politique, F_{RBAC} et $F_{integrite}$ sont fixes une fois le choix du modèle de CA fait. Les ensembles F_{faits} et F_{metier} forment quant à eux le contenu de la politique, *i.e.* ce que les responsables de la sécurité vont administrer.

Les avantages de cette structuration sont multiples. D'une part, nous pouvons identifier les sous ensembles des CTGDs utilisés dans chaque composant, sachant ainsi quels concepts du modèle RBAC peuvent être intégrés avec telle classe de dépendances en connaissant le coût des algorithmes afférents. D'autre part cette structuration permet de faciliter l'administration et l'ingénierie des rôles en définissant des étapes dans la conception de politiques. L'idée de séparer la base des faits RBAC des principes du modèles a été proposée dans [HW03], pour renforcer la séparation intuitive entre la description de la politique et des règles qui la gouverne. Nous avons naturellement prolongé cette approche par un découpage plus fin.

4.2 Base de Faits RBAC

La notion de base des faits RBAC a été étudiée dans les travaux [GB98, SCFY96] : elle représente l'ensemble des faits connus dans la politique. Ainsi, on stocke dans la base des faits RBAC l'ensemble des utilisateurs, les rôles qui leurs sont attribués, la hiérarchie de rôles, les permissions accordées aux rôles et toutes les autres informations utilisées dans le CA. Nous n'avons pas inclus d'attributs dans le schéma Entité-Relation de la figure 2 pour des raisons de lisibilité. La base des faits RBAC, F_{faits}, est stockée dans la méta-base du SGBD. Il s'agit d'informations qui ne se rapportent pas directement aux données gérées par le SGBDs mais plutôt de données systèmes. C'est à partir de ces faits que seront prises les décisions d'autorisations. Il s'agit là véritablement d'un schéma auquel seront ajoutées des dépendances pour former la politique RBAC complète.

Les prédicats présents dans la base des faits RBAC sont :

- *Operation* qui représente les opérations

possible sur les ressources : select, update ou create,
- *Resource* qui représente les éléments dont les accès seront contrôlés : tables, vues, procédures, ...
- *Role* qui représente les rôles identifiés dans la politique, cf. figure 3,
- *Session* qui représente les sujets en cours d'utilisation dans le système,
- *User* qui représente les utilisateurs du système,
- *URA* (User-Role Assignment) qui indique quels sont les rôles qu'un utilisateur peut endosser dans ses sessions,
- *PRA* (Permission-Role Assignment) qui indique quels sont les permissions (permission = opération sur une ressource) accordées aux rôles,
- *dSenior* qui indique quels sont les relations d'héritages directes entre rôles,
- *SU* et *SR* qui indiquent respectivement quel est l'utilisateur associé à la session et les rôles qu'il y endosse.

Dans le modèle RBAC, les sujets du CA sont les sessions : les entités numériques qui représentent les utilisateurs au sein du système. Chaque session est reliée à un seul et unique utilisateur, qui peut endosser simultanément plusieurs rôles. Un utilisateur donné peut disposer de plusieurs sessions, par exemple, s'il utilise simultanément différents applicatifs faisant intervenir tout ou partie des rôles qu'il peut endosser. La base de faits $Ffaits$ suivante est un exemple qui affecte un rôle aux utilisateurs `rthion` et `scoulond`, qui donne aux *infirmiers* le droit de *sélection* sur la table des *prescriptions*, aux *medecins* d'*ajouter* et de *modifier* des prespcriptions et qui modélise la *hiérarchie de rôle* de la figure 3.

```
dSenior(medecin, personnelHospitalier)
dSenior(infirmier, personnelHospitalier)
dSenior(specialiste,medecin)
dSenior(generaliste,medecin)
dSenior(chirurgien,specialiste)
dSenior(pneumologue,specialiste)
dSenior(anesthesiste,specialiste)
dSenior(cardiologue,specialiste)

ura(rthion,infirmier)
ura(scoulond,specialiste)

pra(select,tablePrescriptions,infirmier)
```

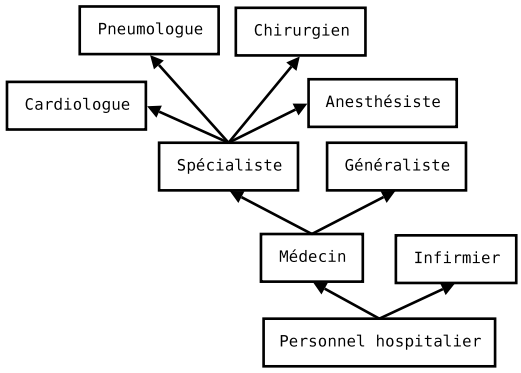

FIG. 3 – Exemple de hiérarchie de rôles, flèche du générique au spécialisé

```
pra(update,tablePrescriptions,medecin)
pra(create,tablePrescriptions,medecin)
```

Notons que certaines implémentations du CA RBAC proposent des variantes du modèle tel que schématisé dans la figure 1. Par exemple, le SGBD Oracle, utilise deux types de rôles : les rôles *utilisateur*, et les rôles *application*. Les principes restent les mêmes que ceux présentés dans cette section, Oracle ajoute simplement un nouvel intermédiaire entre l'utilisateur, et la permission. Les utilisateurs sont affectés aux rôles utilisateurs, qui disposent des permissions par l'intermédiaire des rôles d'application. Cette variante du modèle RBAC est tout à fait représentable dans notre proposition, nous ne l'avons pas utilisée ici pour des raisons de clarté. L'utilisation des outils issus des dépendances pour faciliter la conception de politiques type Oracle est une implémentation que nous envisageons.

4.3 Modélisation des Principes RBAC

Nous supposons que nous disposons d'une base des faits RBAC telle que représentée dans la figure 1. A partir des travaux basés sur la formalisation en programmation logique des modèles RBAC [BS03], nous proposons ici un ensemble de dépendances F_{RBAC} permettant de représenter les principes du CA basé sur les rôles. Nous avons repris les principaux prédicats proposés dans [BS03] ainsi que la modélisa-

(1)	TTGD	$\forall\ Senior, Junior$
		$dSenior(Senior, Junior) \rightarrow senior(Senior, Junior)$
(2)	TTGD	$\forall\ Senior, Interm, Junior$
		$senior(Senior, Interm), dSenior(Interm, Junior) \rightarrow senior(Senior, Junior)$
(3)	TTGD	$\forall\ Ses, Usr, Role, Op, Res$
		$su(Ses, Usr), sr(Ses, Role), pra(Op, Res, Role) \rightarrow authorized(Usr, Op, Res)$
(4)	TTGD	$\forall\ Ses, Usr, Role, JRole, Op, Res\ su(Ses, Usr), sr(Ses, Role)$
		$senior(Role, JRole), pra(Op, Res, JRole) \rightarrow authorized(Usr, Op, Res)$
(5)	TTGD	$\forall\ Usr, Role, Op, Res$
		$ura(Usr, Role), pra(Op, Res, Role) \rightarrow permitted(Usr, Op, Res)$
(6)	TTGD	$\forall\ Usr, Role, Op, Res, JRole$
	$ura(Usr, Role), senior(Role, JRole), pra(Op, Res, JRole) \rightarrow permitted(Usr, Op, Res)$	

TAB. 2 – Dépendances modélisant les principes RBAC, F_{RBAC}

tion de la hiérarchie de rôles. Nous avons étendu cette approche avec la notion de contraintes d'intégrité des politiques, la structuration en quatre parties et l'identification des classes de dépendances utilisées.

La hiérarchie des rôles est une des innovations proposées dans le modèle RBAC. Elle permet de réduire le nombre de rôles en présence dans la politique, simplifiant de ce fait l'administration de la politique et améliorant la productivité des administrateurs. C'est la relation *dSenior* qui représente dans la base des faits RBAC les flèches de la figure 3. Nous avons dans cette figure, huit relations d'héritage direct entre rôles. Par exemple, les *médecins* ainsi que les *infirmiers* disposent de toutes les permissions accordées aux *personnels hospitalier*. La relation d'héritage « complète » est calculée par la fermeture transitive *senior* de la relation *dSenior*. Afin d'exprimer cette fermeture transitive, nous utilisons les deux dépendances (1) et (2) du tableau 2. Grâce à cette fermeture, le rôle *spécialiste*, qui hérite directement du rôle *médecin*, dispose de toutes les permissions accordées au *personnel hospitalier*.

Une fois la relation *senior* calculée, nous pouvons exprimer la *Core Rule* du modèle RBAC, qui s'exprimerait en langage courant par « un utilisateur se voit accordée une permission (une opération sur une ressource) par l'intermédiaire des rôles qu'il endosse dans sa session, par le biais éventuellement de la relation d'héritage ». Cette règle de base est traduite par TTGDs (3) et (4) du tableau 2. Les TTGDs (5) et (6) modélisent le pendant statique de cette *Core Rule*.

Les dépendances exprimant la *Core Rule* ainsi que celles modélisant les propriétés de la hiérarchie de rôle sont des Total-TGDs. Cet ensemble de dépendances F_{RBAC} représente le fonctionnement d'une politique RBAC. Les faits F_{faits} ainsi que l'ensemble F_{RBAC} suffisent pour prendre les décisions de CA. Notons que F_{RBAC} ne comporte que des dépendances de type Total-TGDs, pour lesquelles l'implication est décidable. Nous disposons donc déjà d'une représentation suffisante pour prendre des décisions de CA intégrant la hiérarchie de rôles. Lorsqu'un utilisateur u souhaite effectuer l'opération o sur la ressource Rr, l'accès est autorisé si et seulement si le fait $authorized(u, o, r)$ existe. Ce qui est équivalent à requêter une BD déductive dont la partie en extension (Extensional Database EDB) serait l'ensemble des faits F_{faits} et les règles de production de connaissance, la partie en intension (Intensional Database - IDB), F_{RBAC}.

4.4 Intégrité des Politiques

Gavrila et Barkley ont étudié la spécification formelle des relations entre rôles et entre utilisateurs et rôles [GB98]. Ils ont défini un ensemble de propriétés qui doivent être garanties pour que la politique soit valide. Nous avons repris ces propriétés et les avons exprimées à l'aide de dépendances. Nous traitons donc dans cette section des « contraintes d'intégrité sur la po-

(1)	NGD	$\forall Role\ senior(Role, Role) \rightarrow \bot$
(2)	NGD	$\forall R_1, R_2, U\ ura(U, R_1), ura(U, R_2), senior(R_1, R_2) \rightarrow \bot$
(3)	TGD	$\forall Usr\ users(U) \rightarrow \exists Role\ ura(Usr, Role), role(Role)$
(4)	TGD	$\forall Role\ role(Role) \rightarrow \exists Usr\ ura(Usr, Role), user(Usr)$
(5)	TGD	$\forall Role\ role(Role) \rightarrow \exists Op, Res\ pra(Op, Res, Role), operation(Op), resource(Res)$

TAB. 3 – Dépendances modélisant les contraintes d'intégrité sur les politiques RBAC, $F_{integrite}$

litique ». Les dépendances traitées ici sont des NGDs (avec ou sans contraintes en hypothèse), qui indiquent les états dans lesquels la politique sera inconsistante ($\bot$) :

- aucun rôle n'hérite de lui même : il ne faut pas de cycle dans la hiérarchie des rôles, dépendance (1) du tableau 3,
- aucun utilisateur ne peut être affecté à des rôles héritants entre eux, dépendance (2) du tableau 3,

D'après cet ensemble de dépendances $F_{integrite}$, la politique RBAC sera consistante si et seulement si les faits F_{faits} et des dépendances F_{RBAC} ne produisent pas $\bot$. Il s'agit d'une vérification automatisable qui devra être effectuée avant chaque opération administrative sur la politique, en particulier lors de nouvelles affectations on vérifie qu'aucun utilisateur n'est affecté à deux rôles héritant entre eux et que les contraintes de cardinalité sont respectées.

D'autres contraintes d'intégrité de la politique peuvent être mises en place. En effet, avec la multiplicité des intervenants impliqués dans les grandes organisations, plusieurs administrateurs peuvent se retrouver à collaborer sur une même politique. Afin d'éviter les dysfonctionnements, il peut être intéressant de mettre en place des contraintes imposant un minimum d'affectations, qualifiées de *contraintes de pré-requis*. Ces dépendances font intervenir des variables quantifiées existentiellement. On pourrait dire informellement « qu'elles imposent *l'existence* de certains faits sur des termes inconnus à l'avance ». Nous donnons ici trois contraintes d'intégrité que l'ensemble des faits F_{faits} doit satisfaire :

- chaque utilisateur est associé à au moins un rôle, dépendance (3) du tableau 3,
- chaque rôle est associé à au moins un utilisateur, dépendance (4) du tableau 3,
- chaque rôle dispose d'au moins une permission, dépendance (5) du tableau 3,

4.5 Contraintes Métier sur les Politiques

Nous allons dans ce chapitre aborder ce que nous nommons les « contraintes métier », *i.e.* les contraintes traitées dans la littérature RBAC, dont l'expression dépend des faits de la politique. Il s'agit là de limiter les permissions accordées aux utilisateurs. La littérature distingue deux grandes catégories de contraintes métier : les contraintes *statiques*, qui ne dépendent pas des sessions, et les contraintes *dynamiques*, qui en dépendent. Par exemple, l'exclusion mutuelle entre deux rôles r_1 et r_2 peut être statique « aucun utilisateur ne peut se voir attribuer simultanément les rôles r_1 et r_2 », ou dynamique « aucun utilisateur ne peut avoir de session dans laquelle il endosse simultanément r_1 et r_2 ». Toutes les contraintes métier statiques existent en version dynamique, nous ne présenterons donc que des cas de contraintes statiques. Nous n'aborderons pas les contraintes métier dynamiques qui n'ont pas de pendants statiques (comme celles basées sur l'historique par exemple), dont l'étude est très spécifique.

L'exclusion mutuelle entre rôles est certainement le type de contrainte le plus étudié. La NGD (1) du tableau 4 représente l'exclusion mutuelle entre deux rôles r_1 et r_2. Cependant, si l'on souhaite intégrer les propriétés que doit respecter l'exclusion mutuelle [GB98], il faut prendre en compte la hiérarchie de rôles comme les dépendances (2) et (3) du tableau 4 l'illustrent. Crampton aborde d'autres types de contraintes [Cra03] comme *l'exclusion mutuelle entre permissions* : aucun utilisateur ne doit pouvoir disposer des permissions (op_1, res_1) et (op_2, res_2) (dépendance (4) du tableau 4).

(1)	NGD	$\forall Usr\ ura(Usr, r_1), ura(Usr, r_2) \rightarrow \bot$
(2)	NGD	$\forall Usr, R_1\ ura(Usr, R_1), ura(Usr, r_2), senior(R_1, r_1) \rightarrow \bot$
(3)	NGD	$\forall Usr, R_2\ ura(Usr, R_2), ura(Usr, r_1), senior(R_2, r_2) \rightarrow \bot$
(4)	NGD	$\forall Usr\ permitted(Usr, op_1, res_1), permitted(Usr, op_2, res_2) \rightarrow \bot$
(5)	NGD	$\wedge_{i=1}^{n+1} ura(U_i, r)\ \forall i \in [1..n], \forall j \in [i+1..n+1]\, U_i \neq U_j \rightarrow \bot$
(6)	NGD	$\forall Usr, R\ ura(Usr, guest), ura(Usr, R), R \neq guest \rightarrow \bot$
(7)	NGD	$\forall R, S\ sr(S, guest), sr(S, R), R \neq guest \rightarrow \bot$
(8)	CTGD	$\forall U\ ura(U, chirurgien) \rightarrow \exists U'\ ura(U', anesthesiste), U' \neq U$

Tab. 4 – Dépendances modélisant les contraintes métier, F_{metier}

Les contraintes de *cardinalité* peuvent également être exprimées dans notre cadre, moyennant l'utilisation de contraintes arithmétiques. Nous les considérons comme des contraintes métiers, car elles dépendent des faits de F_{faits}. La dépendance (5) du tableau 4 permet de garantir qu'au plus n utilisateurs sont affectés au rôle r.

Notre cadre nous permet d'exprimer toute une variété de contraintes qui généralisent celles abordées ci-dessus. Par exemple, on peut grâce aux dépendances exprimer une contrainte imposant que « un rôle r soit en exclusion mutuelle avec tous les autres ». Ceci permettrait par exemple, d'interdire l'augmentation des privilèges des utilisateurs invités. La dépendance (6) du tableau 4 exprime cela. La dépendance (7) de ce même tableau est la version dynamique donnée à titre d'exemple. Ainsi, un utilisateur mal intentionné qui à partir d'un compte *invité* parviendrait à obtenir un nouveau rôle rendrait la politique inconsistante et serait remarqué.

Enfin, la dernière dépendance du tableau 4 est un exemple utilisant toute la puissance d'expression des CTGDs (cf. exemple 1 du tableau 1). Il s'agit d'une dépendance utilisant quantificateur existentiel *et* contraintes arithmétiques. Elle exprime le fait que « s'il existe un utilisateur chirurgien dans l'organisation, alors il existe *un autre* utilisateur anesthésiste ». Ce type de contraintes que nous qualifions de *contrainte de pré-requis* n'a pas, à notre connaissance, été jusqu'alors abordé dans la littérature sur le CA. Nous supputons que c'est par manque de pouvoir d'expression des modèles utilisés pour la formalisation des contraintes RBAC.

4.6 Conception et Vérification de Politiques

La mise en place d'une politique de RBAC n'est pas toujours une mince affaire. Comme une politique à base de rôles est « calquée »sur la structure de l'organisation, cela necessite de formaliser cette structure sous forme de rôles, ce qui n'est pas toujours facile et un chantier à part entière. La définition des rôles en présence dans les organisations ainsi que le regroupement des permissions est un domaine de recherche à part entière, à la croisée du management et de la sécurité : l'ingénierie des rôles.

Les contraintes métier telles que nous les avons décrites dans la section précédente pourraient tout à fait être mises en oeuvre techniquement à l'aide des triggers dans un SGBD existant. Cela ne va pas sans poser de problèmes. En effet, pour chaque dépendance que nous avons décrite, il faudrait coder un nouveau trigger. Pour peu que le nombre de faits et de dépendances soit important, cette approche devient impraticable et s'avèrerait certainement très coûteuse. Trouver la cause d'une inconsistance impliquant plusieurs triggers en cascade (ex : pour représenter la transitivité de la hiérarchie de rôle, la *Core Rule* RBAC, les contraintes métier) s'avèrerait très difficile pour les administrateurs. C'est une des motivations pour laquelle nous avons tenté de trouver un cadre permettant la modélisation de toutes les spécificités de RBAC de façon concise et homogène, dans le même modèle que celui utilisé pour l'expression des données que la politique protège.

Les quatre principales étapes de l'établissement d'une politique RBAC ont été identifiées comme étant [FKC03] :

Besoins d'administration	*Réductions aux dépendances*
Vérifier qu'une propriété de sécurité est vraie dans *toute* politique RBAC	Modéliser la propriété sous forme de dépendance f et vérifier son implication $F_{RBAC} \cup F_{integrite} \vDash f$
Eliminer les contraintes métier redondantes	$\forall f \in F_{metier},\ si\ F_{RBAC} \cup F_{integrite} \vDash f$ alors supprimer f car elle est redondante
Interroger la politique	Modéliser la requête sous forme logique, puis trouver les faits unifiables à la requête
Vérifier que la politique est consistante	Utiliser les procédure de preuve pour vérifier que $F_{RBAC} \cup F_{integrite} \cup F_{metier} \cup F_{faits} \nvDash \bot$

Tab. 5 – Réduction en dépendances des besoins d'administration

1. la définition du coeur RBAC, *i.e.* l'identification des utilisateurs, des rôles, des permissions, des affectations utilisateurs-rôles et permissions-rôles,
2. la réorganisation des rôles en hiérarchie à partir des rôles énumérés,
3. la définition des contraintes statiques, avec prise en compte de la hiérarchie des rôles ou pas.
4. la définition des contraintes dynamiques.

Ces différentes étapes font intervenir les différents ensemble de dépendances F_{RBAC}, $F_{integrite}$ et F_{metier} ainsi que les faits F_{faits}. Ainsi, lors de la conception de la politique, différents besoins d'administration doivent être pris en compte :

1. une fois le modèle RBAC choisi, on peut raisonner sur F_{RBAC} et les deux premières contraintes d'intégrité sur la politique $F_{integrite}$ du tableau 3 pour comprendre le fonctionnement du modèle,
2. pendant la définition des faits F_{faits}, on a besoin d'interroger la politique,
3. une fois les affectations faites, on souhaite mettre en place les contraintes de prérequis (les trois dernières dépendances du tableau 3) afin de s'assurer que chacun dispose d'un minimum de permissions,
4. pendant la réorganisation des rôles en hiérarchie, on va avoir besoin de requêter la politique pour trouver les permissions communes à plusieurs rôles,
5. pendant la définition des contraintes métier statiques, on souhaite trouver les faits qui violent les dépendances de F_{metier},
6. une fois les contraintes statiques définies, on souhaite minimiser la taille de F_{metier} afin de simplifier la gestion de la politique,

Chacune de ces problématiques peut être résolue en utilisant les procédures de preuves de l'implication ou la satisfaction logique classique des dépendances. Le tableau 5 résume les grandes classes de problèmes rencontrées dans l'administration des politiques de sécurité et y apporte une réponse en utilisant les outils disponibles dans notre cadre. Le principe est d'exprimer ce que l'on souhaite vérifier dans le cadre des dépendances puis d'utiliser les procédures de preuve pour répondre au besoin d'administration.

Imaginons, par exemple, que les responsables de la sécurité d'une organisation sont en train d'apprendre le fonctionnement du modèle RBAC et souhaitent savoir si dans *toute* politique RBAC*chaque utilisateur dispose d'au moins une permission.* Cette propriété de sécurité s'exprime par la TGD $f \equiv \forall U\ user(U) \rightarrow \exists A, O\ permitted(U, A, O)$. Il faut ensuite vérifier à l'aide d'une procédure de preuve si $F_{RBAC} \cup F_{integrite} \vDash f$. Nous montrerons dans la section 5 comment mettre a profit les étapes de l'inférence des procédures de preuve dédiées aux CTGDs pour *comprendre* comment cette propriété s'avère être vérifiée *pour toute politique* satisfaisant $F_{RBAC} \cup F_{integrite}$, *quels que soient* les faits F_{faits}.

(1)	NGD	$\forall Usr, Role_1, Role_2$
		$ura(User, Role_1), ura(User, Role_2), ssd(Role1, Role2) \rightarrow \perp$
(2)	TTGD	$\forall Role_1, Role_2\ ssd(Role_1, Role_2) \rightarrow ssd(Role_2, Role_1)$
(3)	NGD	$\forall Role\ ssd(Role, Role) \rightarrow \perp$
(4)	NGD	$\forall Role_1, Role_2\ ssd(Role_1, Role_2), senior(Role_1, Role_2) \rightarrow \perp$
(5)	NGD	$\forall Role_1, Role_2, SeniorRole$
		$senior(SeniorRole, Role_1), senior(SeniorRole, Role_2), ssd(Role_1, Role_2) \rightarrow \perp$
(6)	TTGD	$\forall Role_1, Role_2, SeniorRole$
		$ssd(Role_1, Role_2), senior(SeniorRole, Role_1) \rightarrow ssd(SeniorRole, Role_2)$

Tab. 6 – Modélisation de l'exclusion mutuelle et des propriétés afférentes par un prédicat spécialisé

4.7 Prédicats Spécialisés

Pour être cohérente, la politique RBAC doit vérifier un certain nombre de propriétés [GB98]. Dans l'article, les auteurs modélisent l'exclusion mutuelle entre rôles par un prédicat spécifique. Dans notre article, nous n'avons pas préféré spécialiser l'exclusion mutuelle (tableau 4), qui faisait l'objet de symboles de prédicats spécifiques dans [BS03]. Notre parti semble plus générique, dans la mesure où il ne se limite pas un un type donné de contrainte métier. D'un autre côté, il peut s'avérer que la contrainte métier majoritairement utilisée soit l'exclusion mutuelle entre rôles, et que l'utilisation des autres types de contraintes métier s'avère marginale. Auquel cas, il est souhaitable d'utiliser un prédicat spécifique pour représenter l'exclusion mutuelle entre rôles.

Les deux approches ne sont pas incompatibles, en effet, dans les deux cas, nous restons totalement dans le cadre des dépendances. Prenons l'hypothèse que l'exclusion mutuelle nécessite un statut particulier. Nous ajoutons alors un prédicat d'arité 2 $ssd(Role_1, Role_2)$ qui modélise le fait que les rôles $Role_1$ et $Role_2$ sont en exclusion mutuelle (statique). Nous utiliserions alors les dépendances pour modéliser les contraintes l'exclusion mutuelle doit respecter. Le tableau 6 propose un ensemble de dépendances utilisant le prédicat *ssd* [GB98] :

- « un utilisateur ne peut pas être affecté à deux rôles en exclusion mutuelle », dépendance (1) du tableau 6,
- « l'exclusion mutuelle entre rôles est symétrique », dépendance (2) du tableau 6,
- « un rôle ne peut pas être en exclusion mutuelle avec lui même », dépendance (3) du tableau 6,
- « deux rôles en exclusion mutuelle ne peuvent pas hériter l'un de l'autre », dépendance (4) du tableau 6,
- « il ne doit pas y avoir de rôle qui hérite de deux rôles en exclusion mutuelle », dépendance (5) du tableau 6,
- « l'exclusion mutuelle est propagée par la relation d'héritage », dépendance (6) du tableau 6.

5 Validation Expérimentale

Nous avons implémenté en C++ les procédures de preuves CTGDs présentées dans [MS96] avec un prototype appelé (pour l'instant) *CTGD-ToolBox*. Nous avons choisi ce language pour pouvoir utiliser Flex et Bison (ces analyseurs lexicaux et syntaxiques produisent du code C/C++) et pour faciliter l'intégration de notre réalisation avec le SGBD MySQL qui utilise le même language. Notre bibliothèque propose un ensemble de fonctionnalités permettant de manipuler et d'utiliser des dépendances de données de type CTGDs (et donc aussi les sous ensembles de dépendances qu'elles incluent). Grâce à cette boîte à outil nous avons pu réaliser les *chases* présentés en section 3.1 qui nous ont permis de travailler sur les problèmes de vérification des politiques à base de rôles. Actuellement, nous exploitons le prototype pour mettre au point une nouvelle procédure de preuve pour les CTGDs. Ce prototype peut tout à fait s'intégrer dans une interface de conception de BDs,

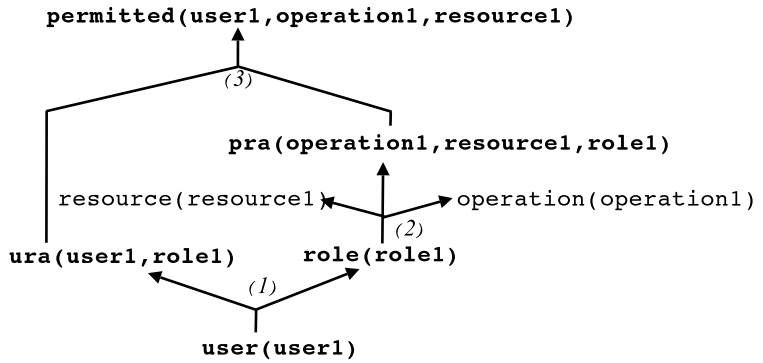

FIG. 4 – Représentation graphique de la trace des procédures dédiées aux CTGDs en marche avant

où il permettrait de s'assurer de la cohérence de schémas ou de la bonne normalisation.

Une des applications que nous proposons [TC06] est une interface d'aide à la conception de politiques de sécurité. Il s'agit d'une interface icônique Microsoft Visio qui permet de concevoir graphiquement des politiques RBAC intégrant hiérarchies de rôles et contraintes statiques. Utilisant le prototype CTGD-ToolBox comme moteur d'inférence sous-jacent, cette interface vérifie en temps interactif que les contraintes sont cohérentes et permet de connaître quelles sont les permissions accordées aux utilisateurs. Nous travaillons sur la prochaine version du prototype où nous porterons notre attention sur l'efficacité et les verrous techniques qui la limite (l'évaluation des contraintes arithmétiques principalement). Les performances actuelles permettent de manipuler en temps interactif quelques centaines de règles, mais ne sont pas comparables à des moteurs de BD déductives comme XSB [CSD]. Intégrer des outils de conception de politique dans les interfaces d'administration des SGBDs est, à notre avis, une proposition importante : une grande partie, si ce n'est la majorité, des failles de sécurité sont dues à des erreurs ou des oublis d'administration. Une telle interface munie des outils que nous avons présentés permet en grande partie de répondre à ce problème en facilitant la conception et l'administration de politiques.

Les procédures de preuves de l'implication dédiées aux CTGDs que nous utilisons produisent des traces exploitables par les administrateurs. En effet, le *chase* n'a pas besoin d'altérer les formules logiques que sont les dépendances, ont évite ainsi la mise sous forme normale de Skolem ou la production de formules intermédiaires dans l'inférence, comme les résolvantes avec la SLD-Resolution par exemple. En effet, les transformations nécessaires seraient :

- transformer chaque CTGD en une formule équivalente en forme prenexe,
- replacer chaque variable quantifiée existentiellement par une fonction,
- séparer chaque CTGD en clauses de Horn.

Avec les procédures de preuves dédiées, chaque étape de l'inférence possède une signification dans la politique RBAC. De plus, comme les formules sont traitées sans découpage préalable, leur nombre n'est pas augmenté. Nous pensons que ces caractéristiques rendent les traces d'inférence exploitables et lisibles, même par des non-logiciens. Pour les administrateurs de sécurité, ceci leur permet de déboguer la politique : de comprendre étape par étape pourquoi la politique est inconsistante, et de pouvoir ainsi y apporter des corrections pertinentes, ce dans le but d'éviter les erreurs d'administration et les dysfonctionnement.

Nous allons illustrer ceci par un exemple.

Reprenant la propriété exposée en section 4.6, on souhaite vérifier que *chaque utilisateur dispose d'au moins une permission*, soit $f \equiv \forall U\ user(U) \rightarrow \exists A, O\ permitted(U, A, O)$. Les administrateurs concevant la politique de sécurité, souhaitent savoir si cette propriété est vraie *quelles que soient* les affectations qu'ils établissement, c'est à dire qu'elle soit impliquée logiquement par les principes du modèle RBAC et par les contraintes d'intégrité établies. Ce problème s'exprime donc par $F_{RBAC} \cup F_{integrite} \models f$. La figure 4 illustre graphiquement les différentes étapes de l'inférence conduisant à prouver qu'effectivement cette propriété est vraie :

1. supposons qu'il existe un utilisateur, identifié arbitrairement par `user1`,
2. par application de la contrainte d'intégrité sur la politique (3) du tableau 3, cet utilisateur est affecté à au moins un rôle, identifié arbitrairement par `role1`,
3. par application de la contrainte d'intégrité sur la politique (5) du tableau 3, ce rôle dispose d'au moins une permission, identifié arbitrairement par l'opération `operation1` sur la ressource `ressource1`,
4. par application de la *Core Rule* (5) du tableau 2, « un utilisateur dispose des permissions des rôles qui lui sont affectés », l'utilisateur `user1` dispose de la permission `operation1` sur `ressource1`,
5. on prouve ainsi que « tout utilisateur dispose d'au moins une permission ». QED.

Nous souhaitons activement mettre à profit notre proposition sur une politique de CA RBAC réelle, mais il s'agit là de données sensibles que les organisations ne désirent pas divulguer pour des raisons évidentes de confidentialité. Utiliser des politiques réelles nous permettrait de valider l'intérêt pratique des différents types de contraintes métier et de savoir s'il est préférable ou non d'utiliser des prédicats spécialisés (pour la modélisation de l'exclusion mutuelle par exemple). Nous avons malheureusement été limités à des exemples de petite taille ou à des jeux de données semi-aléatoires.

6 Conclusion et Perspectives

Nous pensons que la prise en compte nécessaire de la confidentialité des données relationnelles doit se faire de la façon la plus homogène et la plus complète possible. Ajouter des mécanismes dédiés sur les SGBDs afin de répondre techniquement aux besoins des responsables de la sécurité en matière de conception et d'administration de politiques est une source d'erreur et de complexité qu'il est d'après nous préférable d'éviter.

Le domaine des BD propose un large éventail d'outils théoriques et pratiques comme par exemple les dépendances, utilisées pour garantir l'intégrité des données relationnelles ou la conception de schémas par exemple. Il nous paraît nécessaire et profitable de réunir les deux notions de confidentialité et d'intégrité, qui sont souvent traitées indépendamment l'une de l'autre, dans un seul et même cadre théorique. Notre proposition a pour but de mettre a profit la théorie des dépendances pour exprimer et gérer le contrôle d'accès aux données relationnelles. Notre contribution permet d'exprimer le modèle de contrôle d'accès Role-Based Access Control à l'aide des dépendances de données, nous avons ainsi pu prendre en compte les caractéristiques de ce modèle les plus novatrices. De plus, la fertilisation croisée entre les notions d'intégrité et de confidentialité des données nous a permis de proposer un large éventail de contraintes sur les politiques : des contraintes d'intégrité qui permettent de s'assurer de la consistance à l'expression des contraintes métiers. Enfin, nous avons proposé d'utiliser les algorithmes afférents aux dépendances pour assister la conception et l'administration des politiques.

Notre approche offre plusieurs perspectives, allant de la formalisation de propriétés des politiques RBAC en logique à l'aide à la conception. Nous détaillons ici trois sujets de recherche qui étendent la portée de notre proposition.

6.1 Procédure de Preuve en Marche Arrière

Les procédures de preuves pour les CTGDs [MS96] ou pour les TGDs [BV84] permettant d'inférer que $F \models g$, sont typiquement en marche avant, *i.e.* de l'hypothèse de g à sa conclusion. Une procédure de preuve en marche arrière exploitant la notion de *pièce* à été mise au point pour les TGDs [Cou03]. Nous travaillons sur la mise au point d'une procédure en marche arrière pour les CTGDs, en ajoutant la notion de contrainte à la procédure de [Cou03]. Notre prototype *CTGD-ToolBox* intègre déjà la plus part des primitives nécessaires à la réalisation de cette nouvelle procédure. Le prototype nous a déjà permis de mettre en oeuvre la procédure de preuve pour les TGDs en marche arrière. De par leur nature, les procédures de preuve de l'implication en marche arrière peuvent peut-être faciliter la compréhension et l'analyse des politiques de CA.

6.2 Administration de Systèmes d'Information Distribués

La taille considérable des SI actuels, rend l'administration centralisée des politiques de CA absolument impraticable. C'est pour répondre à cette problématique d'administration distribuée qu'on été proposés les modèles d'administration Administrative-RBAC [SM99] et Scoped Administration of RBAC [CL03]. Ces modèles définissent comment donner des privilèges aux différents administrateurs d'une politique RBAC. Nous espérons que les dépendances permettront de représenter ces modèles d'administration, nous pourrons ainsi proposer un ensemble exhaustif de vérifications. On pourrait s'assurer par exemple que les droits donnés aux administrateurs ne conduiront pas à une politique inconsistante.

6.3 Dépendances Génératrices de Tuples Contraintes Disjonctives

Les travaux de [MS96] on étés prolongés par la thèse de Wang [WTM01, Wan02] qui définit un ensemble encore plus général de dépendances de données que les CTGDs : les dépendances génératrices de tuples contraintes disjonctives (Disjunctive Constrained Tuple-Generating Dependencies - DCTGDs). Ces dernières ne restreignent plus les dépendances à des conjonctions d'atomes, elles permettent l'utilisation de disjonctions en conclusion. Nous évaluons l'intérêt de l'introduction de la disjonction dans les règles de CA, elles permettraient par exemple d'exprimer des contraintes métier disjonctives comme « pour tout utilisateur affecté au rôle spécialiste, il existe soit un responsable de fédération de médecine, soit un chef de service (éventuellement les deux) ».

6.4 Contraintes Spatio-Temporelles

Plusieurs proposition ont abordé le problème de l'intégration du temps (voire de l'espace) dans le CA à base de rôles [BS03, JBLG05]. D'un autre côté, l'introduction des contraintes dans les dépendances semble être une manière élégante de pouvoir gérer des contraintes d'intégrité temporelles [BCW95]. Fort de cette idée, nous proposons dans ce chapitre une extension de la notion de contraintes métiers basées sur le temps. Nous ne proposons ici qu'un survol de ce type de contraintes. Supposons que l'on souhaite interdire l'utilisation d'un rôle r sur l'intervalle $[h_1, h_2]$ et que l'on dispose dans le SGBDs d'un système d'horodatage des accès. On peut exprimer cette contrainte avec une dépendance de la forme $sr(S, r), time(T), T \geq h_1, T \leq h_2 \rightarrow \perp$.

Références

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[AS00] Gail-Joon Ahn and Ravi S. Sandhu. Role-based authorization constraints specification. *ACM Trans. Inf. Syst. Secur.*, 3(4) :207–226, 2000.

[Bar02] Steve Barker. Deductive database security. In Ehud Gudes and Sujeet Shenoi, editors, *DBSec*, volume 256

of *IFIP Conference Proceedings*, pages 103–114. Kluwer, 2002.

[BCFP03] Elisa Bertino, Barbara Catania, Elena Ferrari, and Paolo Perlasca. A logical framework for reasoning about access control models. *ACM Trans. Inf. Syst. Secur.*, 6(1) :71–127, 2003.

[BCW95] Marianne Baudinet, Jan Chomicki, and Pierre Wolper. Constraint-generating dependencies. In Georg Gottlob and Moshe Y. Vardi, editors, *ICDT*, volume 893 of *Lecture Notes in Computer Science*, pages 322–337. Springer, 1995.

[BD03] Steve Barker and Paul Douglas. Rbac policy implementation for sql databases. In Sabrina De Capitani di Vimercati, Indrakshi Ray, and Indrajit Ray, editors, *DBSec*, pages 288–301. Kluwer, 2003.

[BS03] Steve Barker and Peter J. Stuckey. Flexible access control policy specification with constraint logic programming. *ACM Trans. Inf. Syst. Secur.*, 6(4) :501–546, 2003.

[BS05] Elisa Bertino and Ravi S. Sandhu. Database security-concepts, approaches, and challenges. *IEEE Trans. Dependable Sec. Comput.*, 2(1) :2–19, 2005.

[BV84] Catriel Beeri and Moshe Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31(4) :718–741, 1984.

[CG99] Frédéric Cuppens and Alban Gabillon. Logical foundations of multilevel databases. *Data Knowl. Eng.*, 29(3) :259–291, 1999.

[CL03] Jason Crampton and George Loizou. Administrative scope : A foundation for role-based administrative models. *ACM Trans. Inf. Syst. Secur.*, 6(2) :201–231, 2003.

[Cou03] Stéphane Coulondre. A top-down proof procedure for generalized data dependencies. *Acta Inf.*, 39(1) :1–29, 2003.

[Cra03] Jason Crampton. Specifying and enforcing constraints in role-based access control. In *SACMAT '03 : Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 43–50, New York, NY, USA, 2003. ACM Press.

[CSD] State University of New York Computer Science Department. XSB System.

[Cup00] Frédéric Cuppens. *Modélisation formelle de la sécurité des systèmes d'informations, Habilitation à Diriger les Recherches*. PhD thesis, Université Paul Sabatier, Toulouse, 2000.

[FKC03] David Ferraiolo, Richard Kuhn, and Ramaswamy Chandramouli. *Role-Based Access Control*. Artech House Publishers, 2003.

[FSG+01] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3) :224–274, 2001.

[GB98] Serban I. Gavrila and John F. Barkley. Formal specification for role based access control user/role and role/role relationship management. In *ACM Workshop on Role-Based Access Control*, pages 81–90, 1998.

[HW03] Joseph Y. Halpern and Vicky Weissman. Using first-order logic to reason about policies. In *CSFW*, pages 187–201. IEEE Computer Society, 2003.

[JBLG05] James Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, 17(1) :4–23, 2005.

[JM94] Joxan Jaffar and Michael J. Maher. Constraint logic programming : A survey. *J. Log. Program.*, 19/20 :503–581, 1994.

[KPP03] Manuel Koch and Francesco Parisi-Presicce. Visual specifications of policies and their verification. In Mauro

Pezzè, editor, *FASE*, volume 2621 of *Lecture Notes in Computer Science*, pages 278–293. Springer, 2003.

[LM89] Jean-Louis Lassez and Ken McAloon. Independence of negative constraints. In Josep Díaz and Fernando Orejas, editors, *TAPSOFT, Vol.1*, volume 351 of *Lecture Notes in Computer Science*, pages 19–27. Springer, 1989.

[LT04] Ninghui Li and Mahesh V. Tripunitara. Security analysis in role-based access control. In *SACMAT '04 : Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 126–135, New York, NY, USA, 2004. ACM Press.

[MS96] Michael J. Maher and Divesh Srivastava. Chasing constrained tuple-generating dependencies. In *PODS*, pages 128–138. ACM Press, 1996.

[MW00] Michael J. Maher and Junhu Wang. Optimizing queries in extended relational databases. In Mohamed T. Ibrahim, Josef Küng, and Norman Revell, editors, *DEXA*, volume 1873 of *Lecture Notes in Computer Science*, pages 386–396. Springer, 2000.

[OAS05] OASIS. Core and hierarchical role based access control (RBAC) profile of XACML v2.0. extensible access control markup language (XACML) 2.0 specification set, Organization for the Advancement of Structured Information Standards, February 2005.

[Rev95] Peter Z. Revesz. Constraint databases : A survey. In Leonid Libkin and Bernhard Thalheim, editors, *Semantics in Databases*, volume 1358 of *Lecture Notes in Computer Science*, pages 209–246. Springer, 1995.

[RS98] C. Ramaswamy and R. Sandhu. Role-based access control features in commercial database management systems. In *Proc. 21st NIST-NCSC National Information Systems Security Conference*, pages 503–511, 1998.

[SB98] Ravi Sandhu and Venkata Bhamidipati. An oracle implementation of the pra97 model for permission-role assignment. In *RBAC '98 : Proceedings of the third ACM workshop on Role-based access control*, pages 13–21, New York, NY, USA, 1998. ACM Press.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2) :38–47, 1996.

[SM99] Ravi S. Sandhu and Qamar Munawer. The ARBAC99 model for administration of roles. In *ACSAC*, pages 229–240. IEEE Computer Society, 1999.

[TC06] Romuald Thion and Stéphane Coulondre. Modeling and inferring on role-based access control policies using data dependencies. In *17th International Conference on Database and Expert Systems Applications (DEXA 2006), to appear*, 2006.

[Wan02] Junhu Wang. *Exploiting Constraints for Query Processing.* PhD thesis, Griffith University, Brisbane, Queensland, Australia, 2002.

[WTM01] Junhu Wang, Rodney Topor, and Michael Maher. Reasoning with disjunctive constrained tuple-generating dependencies. *Lecture Notes in Computer Science*, 2113 :963–973, 2001.